

Máster Avanzado de Programación en Python para Hacking, BigData y Machine Learning

Programación Avanzada Python

LECCIÓN 01

Tipos de datos, operadores y expresiones

ÍNDICE

Introducción

Objetivos

Tipos de datos de Python

Conclusiones

INTRODUCCIÓN

En este primer capítulo vamos a ver los principales tipos de datos que podemos utilizar en Python viendo algunos ejemplos de usos. Además, aprenderemos a leer recoger datos por pantalla. Por último, veremos los operadores relacionales y lógicos viendo ejemplos de usos para entender su funcionamiento.

OBJETIVOS

Al finalizar esta lección serás capaz de:

- 1 Conocer tipos de datos en Python
- 2 Trabajar con cadenas y listas
- 3 Utilizar los operadores relaciones y lógicos
- 4 Obtener datos por pantalla

Tipos de datos de Python

Números

Int

Float

complejos

```
[1] a = 67  
    b = 3.224  
    c = 3 + 4j
```

Type()

Comprobación del tipo de variable

```
[2] print(type(a))  
     print(type(b))  
     print(type(c))
```

```
<class 'int'>  
<class 'float'>  
<class 'complex'>
```


Textos

Las cadenas de caracteres se representan mediante comillas simples o dobles

```
name = "xyz"  
print("Name:", name)  
print("Type:", type(name))
```

```
Name: xyz  
Type: <class 'str'>
```

Lower y upper

Las cadenas se pueden convertir a minúsculas y mayúsculas utilizando la función `lower ()` y `upper ()` respectivamente.

```
y = "PYTHON"  
x = y.lower()  
print(x)
```

```
python
```

```
u = "python"  
y = u.upper()  
print(y)
```

```
'PYTHON'
```

Concatenando cadenas

Utilizando el operador '+' se pueden concatenar dos cadenas.

```
str1 = "hello "  
str2 = "xyz"  
str3 = str1 + str2  
  
print("String after concatenation:", str3)
```

```
String after concatenation: hello xyz
```

Concatenando cadenas

Utilizando el operador '+' se pueden concatenar dos cadenas.

```
str1 = "hello "  
str2 = "xyz"  
str3 = str1 + str2  
  
print("String after concatenation:", str3)
```

```
String after concatenation: hello xyz
```

Concatenando cadenas

¿Qué ocurre en el siguiente código?

```
x = 50
y = "50"

add = x + y
print(add)
```

Concatenando cadenas

```
----> 4 add = x + y  
      5 add
```

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Solución:

```
z = int(y)  
  
add = x + z  
print(add)  
  
100
```

Índice String

Las cadenas están indexadas desde cero y se puede acceder a cada carácter especificando el valor del índice entre corchetes con el nombre de la cadena

```
str1 = "python"  
str1[4]
```

```
'o'
```

index ()

Se utiliza para obtener la posición o índice de un carácter particular de una palabra pasada por parámetro

```
position = str1.index("h")  
print("position of h in str1:",position)  
  
position of h in str1: 3
```


index ()

Hay dos parámetros especificados con la función index (), que son opcionales:

- *start* - establece un índice de inicio para comenzar la búsqueda y su valor por defecto es cero
- *end* - establece un índice final para detener la búsqueda, por defecto es el final de la cadena

string_name . index (value, start, end)

```
string = "ab cd ab"  
position = string.index("a",4,7)  
  
print(position)
```

6

Listas

La lista es un conjunto de elementos guardados de forma ordenada. Sintaxis:

List_name = [item1, item2, item 3... item n]

```
pet_list = ['dog', 'cat', 'rabbit']
```

Listas

Cada elemento tiene un índice, que comienza en cero y se puede acceder a los elementos por su posición de índice:

```
print(pet_list[2])  
  
rabbit
```

La lista puede ser modificada asignando el nuevo elemento a la posición del índice que se quiere modificar :

```
pet_list[1] = 'parrot'
```

Listas

También es posible añadir un nuevo elemento a la lista

```
pet_list.append('cat')
```

Cualquier elemento puede ser eliminado de la lista usando la palabra clave ***del***:

```
del pet_list[2]
```

Slicing

El slicing es un proceso de selección de un conjunto de la lista dando 2 condiciones:

- start - índice inicial de la parte requerida, el índice inicial es inclusivo, es decir, estará presente en la lista
- end - índice final de la parte requerida y es excluyente, lo que significa que no estará presente en la lista.

list_name [start index: end index]

```
shop_list[4:6]
```

```
[20, 'tomato']
```

Lectura desde pantalla

La función `raw_input ()` se utiliza para leer una cadena de caracteres o números desde teclado.

```
number = raw_input("Enter a number:")
```

Operadores relacionales

En Python tenemos un tipo de dato más, es el tipo de datos booleano. Tiene 2 posibles valores: True y False

Operador	Nombre	Ejemplo
<code>==</code>	Igual	<code>x == y = False</code>
<code>!=</code>	Distinto	<code>x != y = True</code>
<code>></code>	Mayor	<code>x > y = False</code>
<code>&lt;</code>	Menor	<code>x &lt; y = True</code>
<code>>=</code>	Mayor o igual	<code>x >= y = False</code>
<code>&lt;=</code>	Menor o igual	<code>x &lt;= y = True</code>

Operadores lógicos

Los operadores lógicos se utilizan para conectar las operaciones relacionales.

- And - si ambas operaciones relacionales son verdaderas, devuelve true.
- Or - si cualquiera de las operaciones relacionales es verdadero, entonces devuelve true.
- Not - si la operación es falsa, devuelve true. Si es verdadera, devuelve falso.



CONCLUSIONES

1

Los 3 tipos de datos más utilizados en Python son : numéricos, cadenas y listas

2

En Python podemos trabajar las cadenas como si fuesen array

3

Los operadores lógicos se utilizan para conectar las operaciones relacionales.

MUCHAS GRACIAS POR SU ATENCIÓN



rsanchezi@grupomainjobs.com



Rubén Sánchez Iruela

[linkedin.com/in/ruben-sanchez-iruela-8156799a](https://www.linkedin.com/in/ruben-sanchez-iruela-8156799a)



twitter.com/eiposgrados



facebook.com/eiposgrados



instagram.com/eiposgrados